Test plan:

A [test plan](https://www.geeksforgeeks.org/software-testing-test-plan-estimates-and-strategy/) is a document that consists of all future testing-related activities. It is prepared at the project level and in general, it defines work products to be tested, how they will be tested, and test type distribution among the testers.

## **Types of Test Plans**:

The following are the three types of test plans:

**Master Test Plan:**In this type of test plan, includes multiple test strategies and has multiple levels of testing.

**Phase Test Plan:**In this type of test plan, emphasis is on any one phase of testing. It includes further information on the levels listed in the master testing plan.

**Specific Test Plan:**This type of test plan, is designed for specific types of testing especially non-functional testing for example plans for conducting performance tests or security tests.

**Entry and Exit Criteria:**The set of conditions that should be met to start any new type of testing or to end any kind of testing.

**Entry Condition:**

* Necessary resources must be ready.
* The application must be prepared.
* Test data should be ready.

**Exit Condition:**

* There should not be any major bugs.
* Most test cases should be passed.
* When all test cases are executed.

Follow these six steps to create an efficient test plan:

1. Define the release scope
2. Schedule timelines
3. Define test objectives
4. Determine test deliverables
5. Design the test strategy
6. Plan test environment and test data

1: Define the release scope

Before any test activity occurs, it’s important to define the scope of testing for your release. This means defining the features or functions that need to be included in the release, considering any constraints and dependencies that can affect the release, and determining what type of release it is.

Examples of questions to ask when defining the release scope include:

* Are there new features being released in this version?
* What are the risk areas?
* Are there any particularly sticky areas where you’ve seen regressions in the past?
* What type of release is it? Is this a maintenance release that includes bug fixes? Is this a minor feature release? Is this a major feature release?
* What does being “done” actually look like for your team?

defining the scope ensures that accurate information is being shared and that there is a common understanding of the product’s goals, expectations, and features.

2. Specify release deadlines to help you decide your testing time and routine. Here are some pointers for determining timelines:

* Consult your project manager to understand the current release timeline.
* Look at past release times and schedules.
* Consider extraneous elements: Does the release need to coincide with outside variables, such as conferences or events? Factor those into your release date prediction.
* Consider the timeframes for development: Your development team might have a set schedule for finishing development work. Make sure you comprehend that timeframe so you can adjust the testing schedule.
* Add some extra wiggle room: It’s common to encounter unexpected delays. Including extra time for unforeseen events can help you stick to your plan.
* Review and update the schedule frequently to ensure the test timetable is attainable.

### 3. Define test objectives

### It is a reason or purpose for designing and executing a test. These objectives ultimately help guide and define the scope of testing activities.

### Examples of general test objectives include:

* Identifying and reporting defects
* Testing new features
* A certain level of test coverage
* **Functional testing objectives:** Ensure the software works as it should.
* Examples of goals for this objective include Validating user workflows, data processing, and verifying input/output parameters.
* **Performance testing objectives:** Ensure the software is efficient and can handle various loads.
* Examples of goals for this objective include Verifying software reaction time, throughput, and scalability.
* **Security testing objectives:**Uncover program security flaws. Examples of goals for this objective include Verifying authentication and authorization features and identifying potential threats.
* **Usability testing objectives:** Concentrate on ease of use and user experience.
* Examples of goals for this objective include Validating software accessibility, verifying user flow, and identifying user-related issues.

### 4. Determine test deliverables:

### Test deliverables are the products of testing that help track testing progress. Deliverables should meet your projects and client’s needs, be identified early enough to be included in the test plan and be scheduled accordingly.

### 5. Design the test strategy:

### Design the test strategy test cost, test effort and which features will be in scope vs out-of-scope.

### 6. Plan the test environment and test data

Planning a test environment guarantees precise and robust testing. The test environment includes hardware, software, and network configurations for software testing. Follow these procedures to set up the test environment:

* **Determine your hardware and program requirements:**Select test environment devices and software, including operating systems, browsers, databases, and testing tools.
* **Install the required software:**Once prerequisites are established, install the necessary tools on the test environment. This may require setting up a separate server to host the application and installing a database management system or other tools.
* **Configure the network:**Make sure that firewall protocols, IP addresses, and DNS settings, among other network configurations, are identical between the test and production environments.
* [**create**](https://www.datprof.com/solutions/what-is-test-data/#:~:text=Data%20can%20be%20created%201,from%20an%20existing%20production%20environment.) **the test data:** Prepare the test material for the application’s testing. Test data can be created manually with data from the production environment, retrieved from an existing production environment and database, or, created via automated Data Generation Tools.
* **Access the builds:**Ensure that the builds that the testers will be testing are accessible. One example is setting up a file-sharing or version control system to allow testers access to the most current builds.
* **Verify the test environment:**After setting it up, check that your test environment fulfills the requirements.

### 

### 